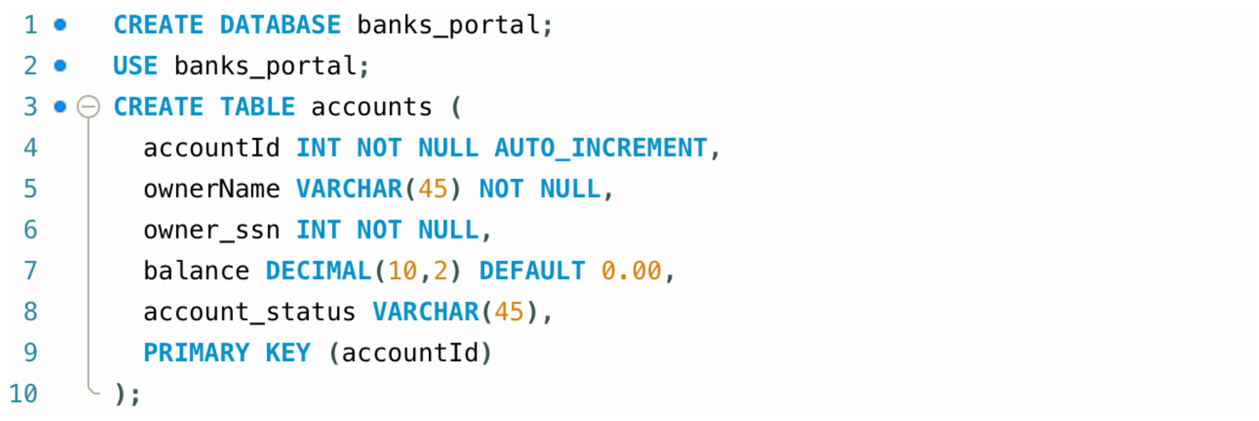
Mahdi Chowdhury
CIS344
Professor Yanilda Peralta Ramos

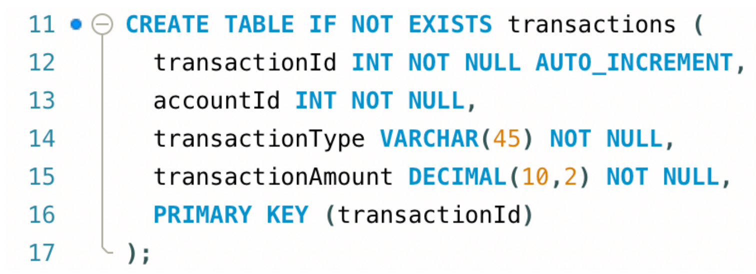
Report

Introduction: This report aims to provide an overview of the development of a banks portal using MySQL, a popular relational database management system (RDBMS). The banks portal serves as an online platform for customers to access banking services, manage their accounts, and perform various financial transactions securely. MySQL is chosen as the database system due to its robustness, scalability, and flexibility in handling large volumes of data in a secure manner.

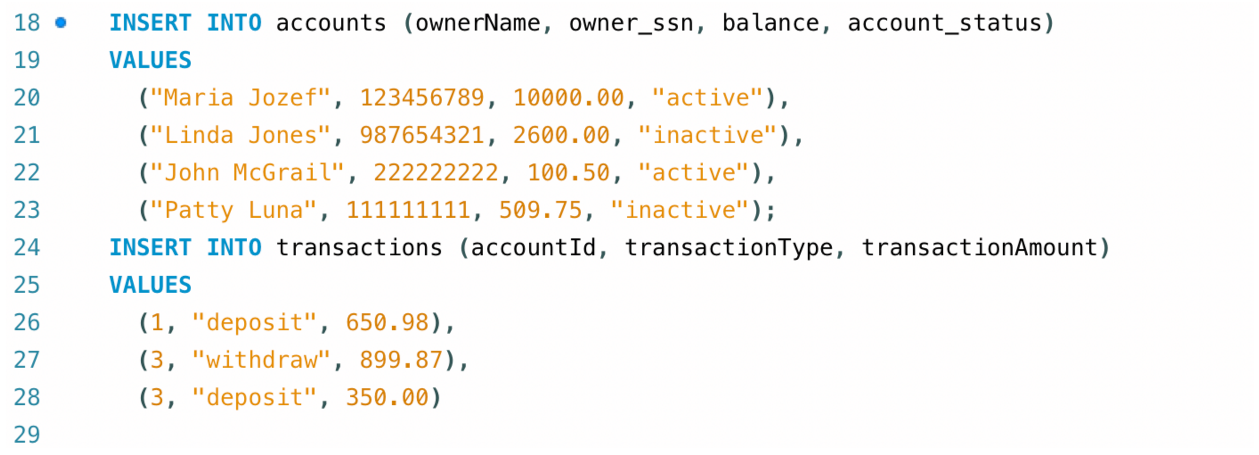
Database Design: The first step in developing the banks portal is designing the database schema. The database design includes defining the tables, their attributes, relationships, and constraints. The main tables in the banks portal database may include 'Customers', 'Accounts', 'Transactions', and 'status'. The 'Customers' table stores customer information such as names, addresses, contact details, and authentication credentials. The 'Accounts' table maintains account-related data such as account numbers, ssn number, balances, types, and associated customer IDs. The 'Transactions' table records all financial transactions made by customers, including details such as transaction IDs, dates, amounts, and related account information. The other sections are withdraw, deposit, delete,account,etc.



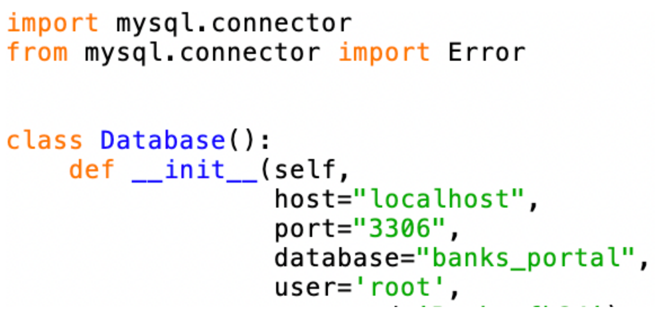
```sql
CREATE DATABASE banks_portal;
USE banks_portal;
CREATE TABLE accounts (
  accountId INT NOT NULL AUTO_INCREMENT,
  ownerName VARCHAR(45) NOT NULL,
  owner_ssn INT NOT NULL,
  balance DECIMAL(10,2) DEFAULT 0.00,
  account_status VARCHAR(45),
  PRIMARY KEY (accountId)
);
```



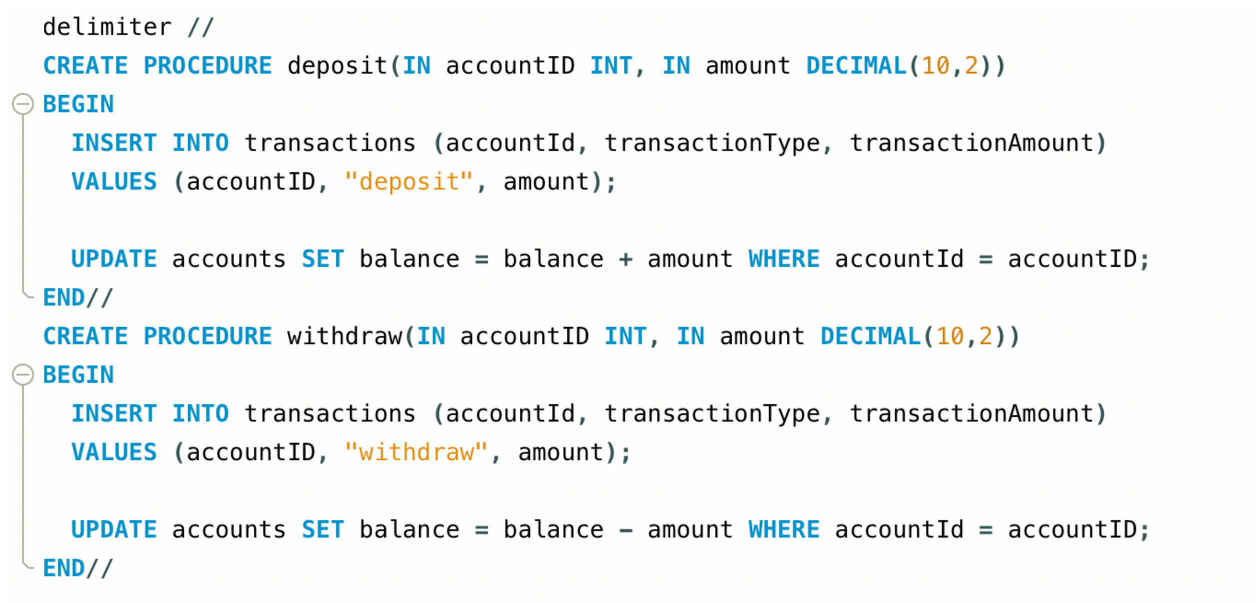
```sql
CREATE TABLE IF NOT EXISTS transactions (
  transactionId INT NOT NULL AUTO_INCREMENT,
  accountId INT NOT NULL,
  transactionType VARCHAR(45) NOT NULL,
  transactionAmount DECIMAL(10,2) NOT NULL,
  PRIMARY KEY (transactionId)
);
```



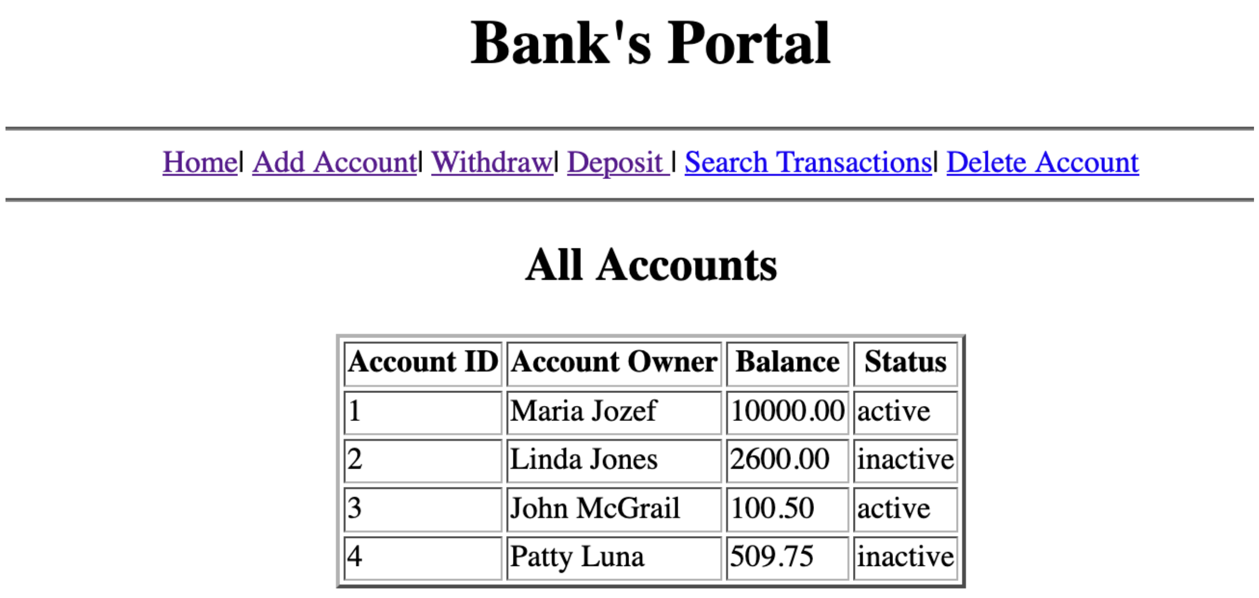
```sql
INSERT INTO accounts (ownerName, owner_ssn, balance, account_status)
VALUES
  ("Maria Jozef", 123456789, 10000.00, "active"),
  ("Linda Jones", 987654321, 2600.00, "inactive"),
  ("John McGrail", 222222222, 100.50, "active"),
  ("Patty Luna", 111111111, 509.75, "inactive");
INSERT INTO transactions (accountId, transactionType, transactionAmount)
VALUES
  (1, "deposit", 650.98),
  (3, "withdraw", 899.87),
  (3, "deposit", 350.00)

```



```python
import mysql.connector
from mysql.connector import Error


class Database():
    def __init__(self,
                 host="localhost",
                 port="3306",
                 database="banks_portal",
                 user='root',
```



```sql
delimiter //
CREATE PROCEDURE deposit(IN accountID INT, IN amount DECIMAL(10,2))
BEGIN
  INSERT INTO transactions (accountId, transactionType, transactionAmount)
  VALUES (accountID, "deposit", amount);

  UPDATE accounts SET balance = balance + amount WHERE accountId = accountID;
END//
CREATE PROCEDURE withdraw(IN accountID INT, IN amount DECIMAL(10,2))
BEGIN
  INSERT INTO transactions (accountId, transactionType, transactionAmount)
  VALUES (accountID, "withdraw", amount);

  UPDATE accounts SET balance = balance - amount WHERE accountId = accountID;
END//
```



Bank's Portal

Home| Add Account| Withdraw| Deposit | Search Transactions| Delete Account

All Accounts

Account ID	Account Owner	Balance	Status
1	Maria Jozef	10000.00	active
2	Linda Jones	2600.00	inactive
3	John McGrail	100.50	active
4	Patty Luna	509.75	inactive

Explanation of all code: This statement creates a new database with the specified name. The USE statement selects the specified database for use. Then statement creates a table with the specified attributes and their data types. The AUTO_INCREMENT keyword ensures that the accountId values are automatically generated for each new row. The IF NOT EXISTS clause ensures that the table is created only if it doesn't already exist. This statement inserts four rows into the "accounts" table, each containing the respective values for the specified columns. After that statement inserts three rows into the "transactions" table, each containing the respective values for the specified columns. This code defines a stored procedure that accepts an accountID and amount input parameters. It uses an INSERT INTO statement to add a new row to the "transactions" table with the provided values. Then, it uses an UPDATE statement to increase the account's balance in the "accounts" table by the specified amount. The last code defines a stored procedure that accepts an accountID and amount input parameters. It uses an INSERT INTO statement to add a new row to the "transactions" table with the provided values. Then, it uses an UPDATE statement to decrease the account's balance in the "accounts" table by the specified amount.

MySQL: provides robust data management capabilities for the banks portal. It supports efficient querying, indexing, and retrieval of data, allowing for quick access to customer information, account details, and transaction records. It has built in security features like admin can control access, encryption and keeping safe the sensitive financial data. Access to the database is granted only to authorized personnel, and different levels of access rights can be assigned based on user roles and responsibilities. Maintaining data integrity is crucial for the banks portal to ensure accurate and reliable financial information. MySQL offers various mechanisms for enforcing data integrity, such as primary key and foreign key constraints, which prevent invalid or inconsistent data from being entered into the database. Additionally, MySQL supports data validation through data types and constraints, enabling the enforcement of rules such as minimum and maximum values, format checks, and uniqueness of certain fields. As the banks portal is expected to handle a large volume of data and serve a growing number of users, scalability and performance are critical considerations. MySQL's ability to handle high data volumes, optimize query execution, and support replication and clustering makes it a suitable choice for a scalable and high-performance database solution. Indexing and query optimization techniques can be implemented to enhance the speed and efficiency of database operations.

Conclusion: Developing a banks portal using MySQL as the underlying database management system offers a solid foundation for building a secure, reliable, and scalable platform for banking services. The database design, data management, security features, and performance optimization capabilities provided by MySQL contribute to the overall functionality and success of the banks portal. By leveraging the power of MySQL, banks can deliver efficient and seamless online banking experiences to their customers while ensuring the integrity and security of their financial data.

Link

https://github.com/mahdichowdhury69/CIS344